Manuel de référence du Logiciel OCAPI Version 2.0

Sabine Moisan, Régis Vincent, Monique Thonnat, Véronique Clément, John van Den Elst

▶ **To cite this version:**

Sabine Moisan, Régis Vincent, Monique Thonnat, Véronique Clément, John van Den Elst. Manuel de référence du Logiciel OCAPI Version 2.0. [Rapport Technique] RT-0183, INRIA. 1995, pp.77. inria-00069988

HAL Id: inria-00069988
https://inria.hal.science/inria-00069988

Submitted on 19 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Manuel de référence du Logiciel OCAPI Version 2.0

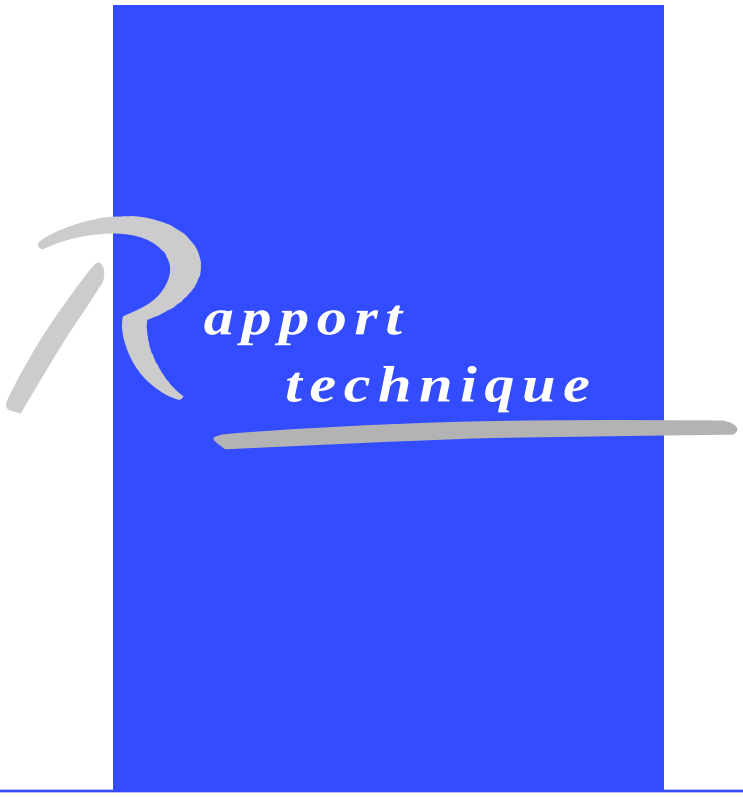
Sabine Moisan, Régis Vincent, Monique Thonnat,

Véronique Clément, John van den Elst

N° 0183

Novembre 1995

PROGRAMME 3



ISSN 0249-6399

Manuel de référence du Logiciel OCAPI
Version 2.0

Sabine Moisan, Régis Vincent, Monique Thonnat,
Véronique Clément, John van den Elst

Programme 3 — Intelligence artificielle, systèmes cognitifs
et interaction homme-machine
Projet Orion

Rapport technique n° 0183 — Novembre 1995 — 77 pages

Résumé : Ce document est composé de quatre parties : la première intitulée **Présentation introductive** décrit les principaux concepts intervenant dans OCAPI et leurs rôles ; la seconde intitulée **Composants d'une base de connaissances** approfondit ces notions et leur utilisation pratique ; la troisième **Moteur et base de faits** présente plus particulièrement le fonctionnement du moteur d'OCAPI et la quatrième **Méthodologie d'utilisation** propose une aide méthodologique à l'utilisateur d'OCAPI pour construire une base de connaissances.

L'interface du logiciel OCAPI est décrite dans le rapport technique [MPT+95]

Mots-clé : Pilotage de programmes, Système à base de connaissances

(Abstract: pto)

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

OCAPI Reference Manual
Version 2.0

Abstract: This document is composed of four parts: the first one entitled **Présentation introductive** describes the main concepts of OCAPI and their roles. The second part: **Composants d'une base de connaissances** deepens these notions and their practical use. The third part: **Moteur et base de faits** focuses on the functionning of OCAPI's engine and the fourth one: **Méthodologie d'utilisation** provides a methodological guide for the OCAPI user to build a knowledge base.

The use of the graphical interface of OCAPI is decribed in a technical report [MPT+95]

Key-words: Program Supervision, Knowledge-Based System

Chapitre 1

Présentation introductive

Ocapi (*O*util de *C*ontrôle *A*utomatique de *P*rocédures *I*mages) est un générateur de systèmes experts, spécialisé dans le pilotage de programmes, initialement conçu pour le domaine du traitement d'images. Il est capable de piloter de manière autonome toutes les étapes nécessaires à un traitement (par exemple l'analyse des images) et de contrôler les résultats obtenus.

La première version d'Ocapi [Clé90] a été originellement conçue en visant plus particulièrement des applications en traitement d'images. Cet outil a été utilisé dans différents domaines : traitement automatique d'images de galaxies [CT91, CT93c, TCO95], ou détection d'obstacles dans un environnement routier [vdE92, TCvdE94]. Ces expériences ont permis de raffiner le modèle [CT93b], qui a été étendu [Ma90, Ala91, Vin93], pour aboutir à la version présentée dans ce rapport.

Un système expert contient la connaissance d'un expert qui développe les bases de connaissances et est utilisé par un utilisateur final qui choisit le contexte et les données d'entrée.

1.1 Description générale

Ocapi est un outil en pilotage de programmes. C'est un générateur destiné à la planification et à l'exécution de programmes d'une bibliothèque pre-existante, pour automatiser le plus possible les différentes phases qui sont nécessaires pour atteindre un certain but.

Ocapi a été conçu selon l'architecture classique à trois composants :

– un moteur indépendant de toute application,

- une base de connaissances développée pour un certain domaine d'application,
- une base de faits dépendante de l'application courante.

Le terme de "*moteur*" est utilisé pour deux composants différents d'Ocapi :

- le moteur de règles qui sert à évaluer les règles de la base de connaissances, c'est le "*moteur*" du générateur dans le sens classique ; c'est un moteur d'ordre zéro en chaînage avant ;
- le moteur de pilotage qui lance automatiquement les programmes pour résoudre un problème. Ce moteur a deux fonctionnalités principales :
  - la planification pour automatiser la sélection des opérateurs ; elle permet de trouver un plan (séquence hiérarchique d'exécution des opérateurs) approprié,
  - le contrôle d'exécution qui sert à initialiser les paramètres, à évaluer les résultats des programmes exécutés et à ajuster ces paramètres en bouclant éventuellement jusqu'à obtenir des résultats d'une qualité satisfaisante.

Une fois qu'une base de connaissances a été créée, le moteur de pilotage de programmes peut exécuter les différentes phases du traitement automatiquement. La planification consiste à choisir entre différents *opérateurs* et à décider de l'ordre d'exécution de ces opérateurs pour atteindre un certain *but* dans un *contexte* défini. Cet algorithme principal est décrit au paragraphe 3.1.1. La planification (donc le choix et l'ordre des opérateurs) dépend du contexte, du but donné et du cas particulier (par exemple une image) à traiter. Les *opérateurs*, les *buts* et le *contexte* sont décrits dans la section suivante.

1.2 Représentation des connaissances

La connaissance est représentée sous forme de *buts*, de *requêtes*, d'*opérateurs* (élémentaires et complexes) et de *règles*. Ocapi fournit deux formalismes différents pour représenter deux types de connaissance différents :

1. des objets structurés pour la connaissance descriptive,
2. des règles de production pour la connaissance déductive.

La connaissance descriptive se compose de toutes les informations disponibles sur les programmes qui seront exécutés automatiquement par Ocapi, sur les liens entre les différentes étapes de traitement, etc. Ce type de connaissance sera représentée en Ocapi par des *objets structurés.*

Les experts humains possèdent de plus une connaissance qui leur permet de choisir un opérateur approprié, d'initialiser et/ou d'ajuster la valeur de certains arguments (en fonction du contexte, du comportement des opérateurs et de résultats intermédiaires) et enfin d'évaluer les résultats obtenus. Ce type de connaissance présente plutôt un aspect déductif ou inférentiel. Elle peut être exprimée facilement dans un formalisme appelé *règle de production.*

1.2.1 Les objets

- Un *but* représente une fonctionnalité du domaine d'expertise (traitement d'images, traitement du signal, ...). Celle-ci pourra être réalisée par un algorithme ou par un traitement plus complexe.

- Un *opérateur* contient la connaissance spécifique pour résoudre un but donné. En Ocapi il y a deux types d'opérateurs : des *opérateurs élémentaires* qui décrivent les programmes de la bibliothèque et les *opérateurs complexes* , qui sont des décompositions en sous-étapes. Les opérateurs complexes permettent d'exprimer la combinaison et le séquencement d'étapes plus simples, afin d'obtenir un traitement complexe.

- Aux buts et opérateurs, sont associés des *arguments d'entrée et de sortie.* Leur description fait également partie de la base de connaissances (*par exemple une valeur numérique comprise entre 0 et 1*). Une requête pourra fixer par exemple des valeurs spécifiques à ces arguments : *image-entree = "g17", seuil = 0.3.*

- Le *contexte* d'utilisation fournit des informations symboliques concernant soit les données (pour une image, par exemple les conditions d'acquisition, le domaine d'application et même des informations sémantiques sur le contenu de la scène) soit l'environnement d'utilisation (par exemple mode automatique ou interactif).

- La demande de résolution d'un problème s'exprime sous forme de *requête.* Une requête précise quel est le but à atteindre, les objets du cas particulier à traiter (par exemple, la désignation de l'image) et la qualité des résultats qu'il faut obtenir.